A5 contd.

of all of its WCs 462 and RCs 464; and for each entry of the WCs 462 and RCs 464, the

IO7 400 relinquishes ownership of each entry. –

On page 13, the first full paragraph:

A6

-- For each of its TLBs 466, the IO7 400 preferably includes a translation invali-

date all (TBIA) register. If the TBIA register contains any value, including all zeros, the

IO7 400 preferably responds by flushing the contents of the respective TLB 466. Ac-

cordingly, as part of the hot swapping of an EV7 processor 202, the user also writes any

value to each TBIA register of the affected IO7 400. In response, the IO7 400 invalidates

the contents of all of its TLB's and relinquishes ownership of each TLB entry. --

REMARKS

The above changes are corrections of errors that were detected after the application was signed by the inventors. This application was "carved out" of a large provisional application, and some of the corrections made by this preliminary amendment reflect this "carving out." However, these differences are of typographical nature and/or use of more precise words, but do not introduce new matter or affect enablement or the claims.

Please charge any additional fee occasioned by this paper to our Deposit Account No. 03-1237.

Respectively submitted,

Edwin H. Paul

Edwin H. Paul, Reg. No. 31,405
Cesari and McKenna, LLP
88 Black Falcon Ave.
Boston, MA 02210-2414
(617) 951-2500

**MARK-UP PAGES FOR THE AUGUST 30, 2001, AMENDMENT
TO U.S. PATENT APPLICATION FILED HEREWITH**

On page 3, the first full paragraph:

-- The present invention provides a method for programmably allocating system

resources to accommodate I/O transactions at I/O ports of a multiprocessor computer

system. The inventive method determines the number and type of transactions antici-

pated at a port, the number and type of devices being serviced via the port, a criteria for

the transactions at the port with respect to the number and type of transactions and <u>de-</u>

<u>vices</u> [device], and assigns the system resources to the port with respect to the criteria. In

preferred embodiments the criteria may include, among other parameters, increasing

system operating speeds, reducing latency, or ensuring that some devices, even low pri-

ority devices, are serviced, slowing the system to allow debugging and other such serv-

icing, ensuring proper communications credits, and supporting hot swapping of processor

and module assemblies. --

On page 9, the second paragraph:

-- In particular, for each port P0-P3 of the IO7 there is a POx_CTRL control reg-

ister 800 of Fig. 7 having a plurality of fields. Figs. 11A-C are a detailed preferred for-

mat of a the POx_CTRL control register 1500. A POx_CTRL control register 800 is

preferably disposed at each port P0-P3 of the IO7 and is <u>written or set-up</u> [read] during

initialization of the IO7. It may also be <u>written or set-up</u> [read] during operation of the

IO7. As shown, the POx_CTRL control register format 1500 is organized into a plurality

of fields. An RM_TYPE field 1502 (Fig. 10C) is preferably used, at least in part, to con-

trol a novel pre-fetch algorithm, which is disclosed in a co-pending patent serial

no._____, entitled, Adaptive Data Prefetch Prediction Algorithm, filed

_____. Which application is hereby incorporated herein by reference.

In particular, the RM_TYPE field 1502 controls the maximum number of DMA engines

that may be assigned to process a given transaction. The RM_TYPE field 1502 may be

2-bits long. --

On page 10, the second full paragraph:

-- The IO7 400 utilizes a credit-based flow control system to communicate with

its respective EV7 processor 202. In particular, for each of the Read I/O, Write I/O, Re-

quest, Block Response and No Block Response virtual channels, the MUX has a corre-

sponding credit buffer. If the MUX has a packet to be transmitted on the Request chan-

nel, it first checks to see if there is at least one credit in the Request channel credit buffer

against which to "charge" this packet. If a credit exists, then the IO7 knows [assumes]

that the EV7 processor 202 has sufficient buffer space to store the packet. Accordingly,

the MUX decrements the Request channel credit buffer by "1" (i.e., the number of mes-

sages to be sent) and sends the message. If there are no Request channel credits, the

MUX must wait until at least one Request credit is received from the EV7 processor 202

before sending the message. --

On page 12, the paragraph starting with "Specifically":

-- Specifically, as described above, each IO7 400 includes a POx_CTRL control

register 1500 (Figs. 11A-C) that contains information utilized by the IO7 400 when it is

initialized. The POx_CTRL register 1500 preferably includes a UPE_ENG_EN field

7

1504 (Fig. 10C). The UPE_ENG_EN field 1504 preferably includes at least one bit for

each DMA engine at the IO7 400. In the preferred embodiment, each IO7 400 has twelve

DMA engines. Accordingly, the UPE_ENG_EN field 1504 has twelve DMA engine en-

able bits. [When the IO7 400 is initialized, it reads the Pox_CTRL control register 1500,

including the UPE_ENG_EN field 1504 and, among other things, starts and runs a DMA

engine for each DMA engine enable bit that is asserted.] Only UPE engines that are en-

abled in UPE_ENG_EN can be used to process DMA transactions. In this way a user can

program the number of DMA engines (up to some maximum, e.g., twelve) that are en-

abled [started]and run at a given IO7 400. If an EV7 processor 202 is to be hot swapped

a user, operating through system software or firmware, preferably de-asserts all twelve

DMA engine enable bits of the IO7 400 coupled to the EV7 202 that is to be removed.

That is, the user sets all bits of the UPE_ENG_EN field 1504 of the respective

POx_CTRL control register 1500 to "0". In response, the IO7 400 stops allocating DMA

engines for new transactions, thereby stopping the IO7 400 from commencing new trans-

actions. When a DMA engine that was in use is subsequently disabled, it nonetheless

completes the pending or existing transaction(s) that were assigned to it. –

On page 12 the last paragraph that runs onto page 13:

-- In addition to stopping the IO7 400 from initiating any new transactions by dis-

abling its DMA engines, the user also causes any data stored in the IO7's WCs 462, RCs

464 and TLBs 466 to be invalidated, whether that data is coherent or not. To facilitate

this operation, among other reasons, the IO7 400 further includes a POx_CACHE_CTL

register at each port 460, which governs the operation of the WC 462 and RC 464 at that

port 460. Fig. 12 is a schematic block diagram of a preferred format of a

POx_CACHE_CTL register 1700. The POx_CACHE_CTL register 1700 includes a

UPE_FLUSH_CACHE field 1702, which may be 1-bit. If asserted, the

UPE_FLUSH_CACHE field 1702 causes the IO7 400 to flush all coherent and non-

coherent data stored in the WC 462 and RC 464 for that port 460. Accordingly, as part of

the hot swapping of an EV7 processor 202, the user also asserts the flush bit of each

port's cache status and control register. In response, the IO7 400 invalidates the contents

of all of its WCs 462 and RCs 464; and for each entry of the WCs 462 and RCs 464, the

IO7 400 relinquishes ownership of each entry [returns Victim or VictimClean messages

to the directory depending, among other things, on the ownership status of the invalidated

data]. --

On page 13, the first full paragraph:

-- For each of its TLBs 466, the IO7 400 preferably includes a translation invali-

date all (TBIA) register. If the TBIA register contains any value, including all zeros, the

IO7 400 preferably responds by flushing the contents of the respective TLB 466. Ac-

cordingly, as part of the hot swapping of an EV7 processor 202, the user also writes any

value to each TBIA register of the affected IO7 400. In response, the IO7 400 invalidates

the contents of all of its TLB's and relinquishes ownership of each TLB entry [sends

VictimClean messages to the directory 380]. --

9